Toward Enabling Convenient Urban Transit through Mobile Crowdsensing

Garvita Bajaj*, Georgios Bouloukakis†, Animesh Pathak†, Pushpendra Singh*, Nikolaos Georgantas†, Valérie Issarny†
*Indraprastha Institute of Information Technology, New Delhi, India
{garvitab, psingh}@iiitd.ac.in
†Inria Paris-Rocquencourt, France
{georgios.bouloukakis, animesh.pathak, nikolaos.georgantas, valerie.issarny}@inria.fr

***Abstract*—The smart cities of the future are expected to be serviced by advanced, personalized multimodal transit systems, charged with timely transport of citizens. Optimizing routes on such networks is a complex problem, in part due to the fact that simple metrics such as latency by themselves are not sufficient to find the *best* routes. In this paper, we focus on the problem of providing commuters with personalized routes with the most *convenience*. We present our mathematical model of user convenience during a multi-leg journey, and the overview of a middleware for enabling convenient transit (including ensuring acceptable network connectivity to mobile apps) by using crowdsourcing. We also report on initial insights obtained through empirical studies on network connectivity and user-perception of convenience in Delhi, India, and Paris, France.**

I. Introduction

With over 70% of the world's entire population expected to be living in cities by 2050, supporting citizens' mobility within the urban environment is a priority for municipalities worldwide. Although public multi-modal transit systems are necessary to better manage mobility, they are not sufficient. Citizens must be offered personalized travel information to make their journeys more efficient and enjoyable. Notably, such information should not only be objective (e.g., bus timetable, live bus tracking), but crucially personalized since passenger preferences and interests differ (e.g., crowdedness of trains, sociability of the coaches).

Enabling personalized travel information requires solution to a multitude of research problems. On one hand, efficient techniques for mobile participatory sensing providing on-demand sensing information at a large scale are required to create robust mobile distributed systems. The sensing has to be intelligent to allow resource conservation as well as capture of correct sensing information in a given context without requiring user intervention all the time, and in spite of network unreliability [1]. These techniques need to then be complemented by domain-specific machine learning algorithms, which must be able to execute on resource-constrained mobile devices with heterogeneous configurations.

It is important to note that although a large body of work exists focusing on the use of ICT in multi-modal urban transportation in the western world ([2], [3]), the rise of public transit in emerging markets brings unique challenges to the domain. Notably, although it can be assumed that all urban commuters in a country like India have mobile phones, even smart phones, not all might have data plans and might rely on SMS only. The extreme unreliability of the actual transport system due to chaotic large-scale traffic (e.g., buses not following their schedule) further stresses the need of robust systems for information management in these contexts.

Thanks to the increased abundance of mobile phones, the recent field of mobile participatory sensing could be leveraged towards providing a more fine-grain and up-to-date view of a city's transportation system. In this paper, we focus on two different aspects of using mobile participatory sensing for providing personalized transit solutions: First, developing a scalable data collection middleware, that gathers both passive sensory information and active user-generated content, and disseminates transport network status updates to travelers as per their mobility preferences for personalized services. Challenges here include the large scale, system dynamicity, and heterogeneity while combining the data streams of interest to a user. Second, we are developing learning and mining techniques for sustainability, that can help identify events of interest from collected data; and help provide personalized services to complement the first aim. Recently, machine learning has been increasingly used in enhancing capabilities of smartphones. It has been used in detecting everyday activities [4], road conditions and driving behavior [5], inferring information from complex sensors like audio [6] etc. Our work explores different applications of machine learning in providing personalized services and develop new approaches that enable such services in urban transport context.

Unlike existing transit applications that recommend routes to users based on objective metrics like least distance, least hops etc., we aim to provide commuters the most *convenient* personalized routes. *Convenience* is a subjective term and may differ among users. In this paper, we identify three different parameters on which user-convenience depends and develop a city-wide convenience model as a first step towards providing convenience information to passengers. We focus on two parallel paradigms aimed at improving transit services for urban travellers. On one hand, we present our mathematical model of user convenience during a multi-leg journey in Section II, and validate it using data collected from metro users. On the other hand, since transit conditions vary with time, users must

be presented with real-time convenience information provided by other travellers, we aim to identify the best interaction paradigm for enabling component interaction by presenting an overview of the middleware for enabling convenient transit (including ensuring acceptable response times to mobile apps) by using crowdsourcing in Section III. Based on our prototype application (Section IV), we report on initial insights obtained through empirical studies on network connectivity and user-perception of convenience in Delhi, India and Paris, France India in Section V. Section VI concludes with a discussion of the results and directions of our future work.

II. Convenience Model

To improve transit facilities in cities, it is important to understand the user perception about existing conditions. For this, we propose a convenience model – a function of time delay, congestion, and seat availability experienced by passengers of Delhi and Paris. We choose time and congestion as the two parameters, inspired by the work in [2] that used a similar model for developing traffic regulators for metro rails. We also consider seat availability as another important parameter as we believe that comfort level is affected by seating time (also related to crowdedness in the metro). Metro rail behavior is similar across cities; passengers do not know the exact arrival and departure times of metro rails but have an experience of expected waiting and travel times for a path. Similarly, passengers have an expectation of congestion levels and seat availability which helps them decide their ingress and egress stations. Hence, we build convenience models around expected parameter values and not actual parameter values.

The metro network of a city can be represented as a directed graph $G = (V, E)$ where V is the set of stations and E is the set of directed edges such that $(u, v) \in E$ where $u, v \in V$ is a directed edge. We model the inconvenience experienced by a passenger travelling on a path P where P is a series of transit legs $L_1, L_2, ..., L_n$ lying on the path. The transit legs may belong to different lines connected by junction stations, i.e., stations where the line switch happens. Let J denote the set of junction stations and N denote the set of non-junction stations on the path P. Therefore, $P = J \cup N$.

We model the inconvenience caused to a passenger on the path $P_{(o,d)}$ for the vertex pair $(o, d) \in V \times V$ where o is the originating station and d is the destination station. The path $P_{(o,d)}$ is a sequence of vertices $v_1, v_2, ..., v_n$ on transit legs $L_1, L_2, ..., L_m$ with repetitions allowed. In the following text, we model the inconvenience caused to passengers in terms of the identified parameters.

The time delay experienced by a passenger can be divided into the following segments:

1) t_e^t : time taken to travel the edge e
2) t_v^s : stoppage time at vertex v
3) t_v^{sw} : time to switch from one leg to another at junction station v

We make the following network assumptions for our model: *i)* time taken to travel an edge e (t_e^t) is fixed; *ii)* stoppage time at every station $v \in G$ (t_v^s) is fixed; and *iii)* time taken to switch transit legs at a junction station $j \in J$ (t_j^{sw}) is fixed. The total estimated time to travel from o to d E_t is:

$$E_t = \sum_{e \in P} t_e^t + \sum_{v \in N - \{o,d\}} t_v^s + \sum_{j \in J} t_j^{sw} \tag{1}$$

For a timely service, the actual travel time should be equivalent to E_t. However, the traffic of metro type railways is intrinsically unstable, and delays may occur due to the following reasons causing passenger inconvenience: *i)* metro may arrive late at origin (passenger has to wait at station o longer than expected), *ii)* it may run slow causing longer travel times, *iii)* it may have to wait at a station for long, or *iv)* there may be huge crowd at a junction station resulting in longer switching times The time inconvenience caused to a passenger is, therefore, the sum total of all of these. Let *i)* A_o denote the actual waiting time at station o and E_o be the expected waiting time. (E_o can be regarded as the frequency of arrival of metro at station o.); *ii*) A_e^t denote the actual time taken by the metro to travel edge $e \in E$; *iii)* A_v^s denote the actual stoppage time at station $v \in N$; *iv)* A_e^t denote the actual time taken to switch transit leg at junction station $j \in J$; and *v)* A_c represent the actual inconvenience experienced by the passengers.

The inconvenience caused due to the delay in metro service is modelled as:

$$I_t = (A_o - E_o) + \sum_{e \in P} (A_e^t - t_e^t) + \sum_{v \in N} (A_v^s - t_v^s) + \sum_{v' \in J} (A_{v'}^{sw} - E_{v'}^{sw}) \tag{2}$$

To model the congestion inconvenience, we use the concept of personal space defined by Edward Hall's personal reaction bubble [7]. The travel of a passenger is assumed to be inconvenient only if there are other commuters within a passenger's personal space (a radius of 1.5 ft). The congestion inconvenience is, therefore, given by:

$$I_c = \sum_{l \in L} \int_{T_l}^{T_l^s} (A_c - N_c) dt \tag{3}$$

where T_l represents the start of travel on leg l, T_l^s represents the time at which the passenger gets a seat on leg l, and N_c represents the nominal congestion calculated using reaction bubble. The journey rating of a passenger may also vary with the seat availability in the metro. A user may skip a crowded metro hoping for a seat in the next metro. Hence, lack of seat availability may increase the inconvenience experienced by users (represented by I_s and reported directly by users).

Hence, the overall inconvenience caused to a passenger throughout the travel from o to d can be linearly related as:

$$I = aI_t + bI_c + cI_s + \eta$$

where a, b and c are city-dependent parameters defining the weights of inconvenience and η is a city-specific constant.

Since these parameters depend on the city conditions, we developed an Android application – *Metro Cognition* (Section IV) to collect data to identify these variables. Our goal is to define the weights of these parameters for predicting a user's journey experience to provide personalized services. The analysis of the data collected using the app is given in Section V where we summarize our results.

III. System Architecture

Metro Cognition is a mobile application targeted for urban smartphone users traveling by metro trains. Such a user may act as a **sender** (gathering and sending data to a main component), or a **receiver** (accepting data related to metro services for facilitating her travel experience). To enable an effective way of interaction between the users and the software components, we need to take into account three basic constraints: i) *Connectivity* - the connectivity status changes quickly while traveling into the metro; ii) *Energy efficiency* - smartphones are limited in terms of battery availability, and; iii) *Timeliness (Freshness) of data* - for instance, data of a traveler reporting an inconvenience is valid for a short period.

The above constraints pose a barrier for enabling efficient component interactions. For instance, the connectivity status affects the freshness of a message and, sending/receiving a tremendous amount of messages may result into quick battery drain on mobile devices [8]. So far, a number of middlewares have been proposed to tackle common limitations of mobile environments. These can usually be classified into a set of common interaction paradigms. One suggestion to improve the quality of interactions is to enable the system designer to select the appropriate interaction paradigm regarding the constraints and the application context. Relying on our prior experience [9], we classify the interactions of software components into the following paradigms:

- *Client/Server (CS)* - a client communicates directly (without intermediate components) with a server either by direct messaging (one-way) or through remote procedure calls (RPC, two-way).
- *Publish/Subscribe (PS)* - multiple peers interact via an intermediate *broker*. Subscribers subscribe to a specific topic on the broker, publishers produce *events* in that topic and subscribers receive events in FIFO order.
- *Tuplespace (TS)* - multiple peers interact via an intermediate *shared data space*. Writers post *tuples* to the shared data space, and readers can retrieve data from it using a specific template (non-FIFO order).

Focusing on receiver's part and considering the main constraints described above, in the CS interaction paradigm, server always receives *fresh* messages and *battery* reduces by a certain amount when *connectivity* status is *ON*. On the other hand (when the connectivity is *OFF*), messages are lost. In PS interaction paradigm, subscribers receive events regardless of their connectivity status. When the status is *ON*, events are received immediately (*fresh*). When the status is *OFF*, events are stored with the broker and received by subscribers (*stale data*) in a FIFO order when connectivity is restored. The battery is reduced by a certain amount for all events. According to the disconnection period, battery is likely to drain significantly depending the amount of events sent [10]. Finally, in TS interaction paradigm, there is a possibility (when connectivity status is *ON*) to obtain tuples immediately (*fresh*) and battery reduces by a certain amount for each tuple. On the other hand, when the status is *OFF*, tuples are kept to the data space. When connectivity is restored, readers are able to read *stale* tuples based on the template applied. In other words, readers receive a number of tuples (e.g. the most important) and the battery reduces according to these tuples. The latter makes mobile users more flexible to consider the number of retrieved tuples based on the remaining battery.

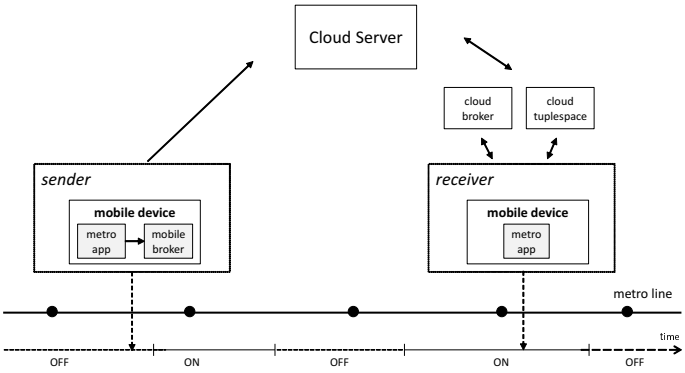


Fig. 1: Metro Cognition user (sender and receiver) traveling in the Metro

The proposed architecture of *Metro Cognition* can be represented using appropriate interaction paradigms towards developing an effective and usable mobile application. As depicted in the Fig. 1, while a user is traveling in the metro, connectivity issues might occur. Also, mobile users in the metro can act as a: i) **sender**: following a PS interaction paradigm, user's data should be gathered with a broker - embedded to her mobile device. According to the connectivity status, data are forwarded to the main software component (CS server) for processing; or ii) **receiver**, data are sent from the main software component (CS server) in order to reach mobile users either through PS, or TS interaction paradigm depending on the remaining energy on the mobile user's battery. The intermediate components (broker, tuple-space), should be independent and deployed in an always-reachable machine (e.g. cloud).

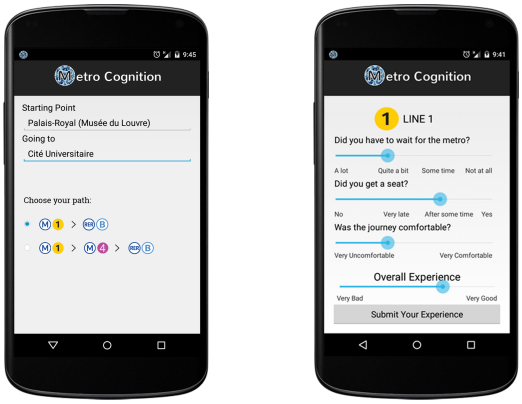
The above interactions should be coordinated by a third-party platform, which we call *Sarathi*[1], dedicated to transportation systems. In this work, we take our first step towards *Sarathi* by implementing *Metro Cognition* for **senders** using the *GoFlow*[2] middleware in order to gather and send data to a server. *GoFlow* follows a Publish/Subscribe interaction paradigm, providing queueing mechanisms to store data while senders are disconnected.

IV. Application

Our goal is to develop services for urban travelers that allow users to specify personal preferences while looking for public transit modes. The input to the service would be a set of personalized user preferences (e.g., wishing to sit during the journey, path with the shortest travel time, etc.), and the output would be a list of public transit modes that best meet

[1]Sarathi is a Hindi word meaning Charioteer - the driver of a Chariot

[2]http://goflow.ambientic.mobi/



(a) Users select their path from a list of options

(b) Feedback form to submit user experience rating

Fig. 2: Screenshots of the mobile app

the user specifications. Towards this end, our first goal is to build models that can predict the convenience level of a user based on the parameters identified in Section II.

We developed an Android application, *Metro Cognition*, to collect ground truth data required for identifying the constants a, b, c and η of the convenience models for Delhi and Paris (screenshots shown in Fig. 2). Unlike the fine grained division in convenience model, we opted to collect coarse grained parameter values on a scale of 4 - *very bad*, *bad*, *good*, and *very good* - keeping the user inputs as low as possible. The collected data are used to train our model (the convenience level experienced by passengers for different parameter levels) using machine learning techniques (see Section V).

Taking into account the basic constraints described in Section III, we collect data in a reliable way by following the proposed architecture. Using the *GoFlow* middleware, we implement our Android application with an embedded broker acting as a sender providing data to the cloud server as depicted in Fig. 1. Moreover, providing services that are QoS (Quality-of-Service) aware (e.g., wishing to sit during the journey) presupposes the awareness of connectivity status of metro users. Towards this, Metro Cognition collects connectivity tuples every 30 seconds using a background service while the user is traveling. Let `con_tuple` be the connectivity tuple with the following 6 values: i) *ON/OFF* (depending on availability of internet); ii) *timestamp* (the exact time when connectivity is ON/OFF); iii) *location* (GPS coordinates); iv) *accuracy* (the location accuracy); v) *provider* (e.g. Vodafone) and; vi) *path_id* (a unique identifier for a user's path). User data is associated with a unique ID - a unique hash representing a user's device so it is not possible to identify users from the data. We also understand that locations collected as part of `con_tuple` might reveal user identity, and we plan to resolve privacy implications as part of our future work.

To use the application, a user selects a path between two stations using the screen shown in Fig. 2a. The user can move the application to the background, generating a notification icon on the status bar to remind users to submit their feedback. When the journey ends, the user can click on the notification to launch the form (shown in Fig. 2b) and submit her feedback. Convenience and connectivity data are stored in JSON format and sent to the *GoFlow* cloud server.

V. Early Experiments

In order to evaluate our approaches, we performed similar experiments in Delhi, India, and Paris, France. The data for the experiments were provided by volunteers traveling in the metro rails of both the cities. For Delhi, we provided an incentive, Rs. 100 for every 50 stations of data, to the volunteers providing the data. It may be noted here that the volunteers were students who traveled by metro to college everyday, and were not asked to take extra journeys for data collection. Volunteers from Paris were colleagues who agreed to provide the data for free. A total of 24 users volunteers participated (12 from each city).

In the following text, we describe our experiments and analysis of the results generated.

A. Convenience Analysis

As described in Section IV, we collect user feedback on three parameters (time delay, comfort experienced, and seat availability) for their metro journey in a subjective manner to build convenience models for the two cities. Our goal is to identify the best machine learning techniques suited for this data, and also identify the weights of different parameters contributing to the journey ratings of travelers (including identification of hidden parameters, if any). Metro journeys may vary in length (single-legged, bi-legged, tri-legged, etc.), and hence, a single model cannot suffice for all trips. Thus, we divide the dataset into different categories and analyse them independently. Table I shows the number of samples collected by volunteers for each category. The diversity in the number of samples in the divided datasets suggests that passengers in Paris take single-legged journeys more often than multi-leg journeys. For Delhi, however, passengers mostly take three-legged journeys. Since the number of data samples for one-leg Delhi journeys, and three-leg Paris journeys is too less, we do not use them for building convenience models. We currently focus on two-leg journeys for Delhi and Paris, three-leg Delhi journeys, and one-leg Paris journeys.

We tried four different classification models - decision trees [11], multiclass linear regression [12], SVM [11] and Neural Network [12] on the dataset. Since the dataset is small, we assumed SVM (Support Vector Machines) and NN (Neural Networks) would yield poor classification models. SVM analysis validated this as more than 30 data samples (out of 40 from Paris one-leg and Delhi three-leg dataset) were being used as support vectors, causing *overfitting*. This resulted in low test accuracy, rendering this technique unsuitable for our dataset. Neural networks, on the other hand, resulted in high accuracies even on small dataset. The high accuracies were attributed to the hidden layer of neurons which considers the combined effect of multiple parameters and adds a bias parameter to the model. We tried MultiLayer Perceptron (MLP) and neural network with Radial Basis Function (RBF). The accuracies

TABLE I: Number of data points provided by volunteers

	One Leg	Two legs	Three legs	Total
Delhi	7	38	53	98
Paris	52	37	15	104

TABLE II: Accuracies (in %) obtained with different neural network settings

	Multilayer Perceptron				Radial Basis Function			
Dataset[3]	D3	D2	P2	P1	D3	D2	P2	P1
Training	97.2	84.6	92.9	89.5	78.6	84.6	91.3	82.8
Testing	83.3	83.3	91.7	80	83.6	81.8	83.3	80.2

achieved over a 80-20 training-testing split of collected data are shown in Table II. MLP resulted in higher accuracy than RBF NN due to the presence of a feedback loop (backpropagation). This suggests interdependence between parameters and assures the suitability of this technique on the dataset.

Since the dataset was small, and learning convenience model involves classification of user data, we also tried decision tree based training. We trained regression decision trees using `fitrtree` function of MATLAB R2014b. The average test accuracies (reported on 2000 iterations of decision trees constructed using 80% data for training) were less than 75% for both Delhi and Paris metro data, and the standard deviations in the reported accuracies were close to 0.15. A high 10% difference in accuracies of neural networks and decision trees suggest the suitability of neural network on our dataset.

Another goal of our app is to identify the relative importance of the parameters in the convenience model. The goal is to provide a list of importance parameters to transit companies and help them improve their services. To weigh the importance, we use multiple linear regression for categorical variables (using IBM SPSS tool [12]) to identify the coefficients of parameters associated with user rating in the convenience model.

We trained different linear regression models using all parameter combinations - *time delay*, *seat availability*, *user comfort*, *time delay* and *seat availability*, *seat availability* and *user comfort*, *user comfort* and *time delay*, and all three together. Comparing the obtained parameter coefficients, *comfort* was the most weighted parameter deciding the user ratings in Paris. A linear regression model applied on Paris one-leg dataset suggests that parameter weights follow the order: *Comfort* $>$ *Seat availability* $>$ *Time delay*. Also, η value is low (less than 1 for multiple parameters) suggesting that noise in the trained model is less. Delhi, however, shows a different behavior towards considered parameters. All parameter coefficient values are small in magnitude showing no preference towards a single parameter, but η values are significantly high (upto 2.06) for all combinations indicating the presence of other hidden parameters contributing to the user ratings. This is also justified by the high accuracy achieved using hidden layer neural network for Delhi dataset. This analysis highlights the difference between metro user experiences in the two cities and validates the need for different convenience models depending on the user experiences.

B. Connectivity Analysis

Metro Cognition also collects data related on the users' Internet connectivity status. Awareness about network issues in the metro will allow our platform to take decisions at runtime to provide services that are adaptable and QoS-aware.

Based on our dataset given in Table I, we analyze the user's connectivity pattern (`con_pattern`) using 98 data points in Delhi and 104 in Paris. One `con_pattern` consists of many `con_tuple` (defined in the Section IV). Each `con_pattern` is created as follows: *i)* the user starts the application and chooses her path, *ii)* the background connectivity service starts, *iii)* when the connectivity status changes, a tuple (`con_tuple`) is stored, *iv)* if the connectivity changes to ON, the background service stores tuples every 30 seconds; *v)* when the user's journey ends, she submits the form and the background service stops.

Note that we do not rely on the Android API for monitoring connectivity changes as it does not monitor all actual disconnections - there are many occasions when the phone might report internet connectivity, while HTTP endpoints on the internet are unreachable. Hence, the results are inaccurate. Instead, we perform an `HTTP GET` to a URL that rapidly returns an `HTTP 204` response to ensure actual 2G or 3G connection availability. For instance, on 5th May, 2015 a user traveled from Cité Universitaire to Porte d'Auteuil with `Free` telecom provider. The total duration of her trip was 32.6 minutes. As per the Android API, the user was connected for 71.4% of the journey, while our approach demonstrates that she was connected only for 29.7% of the overall period.

By observing available connectivity patterns, no repeated patterns were observed, even if we repeated the same path every day at the same time. However, we hypothesize that connectivity patterns depend on crowdedness of the metro. To verify our assumption, we conducted the following experiment: *i)* we picked the largest business districts - Rajiv Chowk and La Défense, and two residential districts, Govindpuri and Cité Universitaire, in Delhi and Paris respectively; *ii)* during a working day, we travelled from one station to another (Govindpuri $\rightarrow$ Rajiv Chowk, Cité Universitaire $\rightarrow$ La Défense) and back (Rajiv Chowk $\rightarrow$ Govindpuri, La Défense $\rightarrow$ Cité Universitaire); *iii)* we classified the routes into 3 categories (*Morning*, *Mid-day*, *Evening*); and *iv)* we provided the percentage of time a user was connected during the trip. Fig. 3 depicts the connectivity over time of the experiment described above for some representative cases. As shown, the overall disconnected time and the number of disconnections are high in the morning when people go to work in La Défense and in the evening when people return home from Rajiv Chowk (crowded metro). On the other hand, since for the remaining parts of the day, there are less movements to these areas, the overall disconnected time and the number of disconnections decrease. Also, according to Table III, people traveling to an office area in morning, and those traveling to a

[3]Dx corresponds to the dataset from Delhi x-legged journeys, and Py corresponds to dataset from Paris y-legged journeys

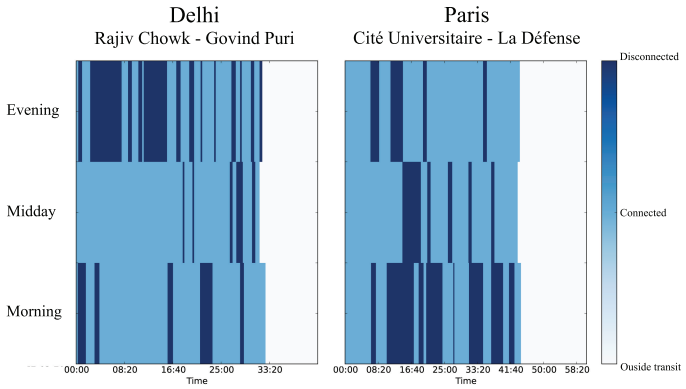


Fig. 3: Connectivity over time

residential one in evening are connected less than 59% during their journey. For the remaining parts of the day, the overall connectivity can reach up to 88%. Since travelers commute often during mornings and evenings, it is important to identify network conditions that can support real-time information dissemination.

VI. Discussion

In this paper, we present our initial idea of enabling convenient urban transit through mobile crowdsensing. We emphasize the importance of user convenience as a metric to improve public transit systems. We collect training data using an Android app and develop convenience models for Delhi and Paris using machine learning techniques. We report the highest accuracies received by neural network training on subsets of data considered and summarize our results.

Although our work is in preliminary stage and the available data used for the training models is small, we can still identify stark differences between user experiences. For Paris, the considered parameters are linearly related to the journey ratings of users. For Delhi, however, the high magnitude of constant term indicates the presence of other parameters (currently unknown) which contribute to the journey rating of passengers. These parameters could be associated with the inconvenience experienced by passengers in reaching metro stations (metro stations in Delhi are located at longer distances), with the sex of the users (women metro riders in Delhi can use a dedicated "Ladies coach"), the time of travel, or weather. We aim to identify these hidden parameters in the context of Delhi users as part of our future work.

To ensure reliable data collection and provide QoS-aware services to mobile users, we provide an overview of our middleware. Our platform, *Sarathi*, takes into account the constraints of mobile environments to satisfy components' interactions by identifying ideal interaction paradigm. Towards this, it is essential to obtain insights on internet connectivity in the metro. We monitor the connectivity changes during multiple journeys in the metro of Paris and Delhi. We show that our approach is accurate and depicts reality in terms of internet connection. Initial study of the users' connectivity patterns suggests that being connected depends on the crowdedness of metros. For our future work, we plan to identify correlation between network conditions and comfort level of users, and plan to use the developed convenience models in an Android app to provide personalized mobility services. Moreover, we intend to utilize connectivity patterns as a realistic input-parameter to our *queueing network*, which is part of our ongoing work. The outcome will allow us to identify the systems' response time and improve it by applying other timing parameters (e.g., setting a time-to-live for each data record). Finally, these initial set of experiments have given us guidelines for executing larger-scale experiments which will provide us statistically significant results.

TABLE III: Percentage of time connected during experimental journey

	Morning	Midday	Evening
Cité Universitaire - La Défense	51%	81%	83.5%
Govind Puri - Rajiv Chowk	59%	88%	76%
La Défense - Cité Universitaire	78%	81%	44%
Rajiv Chowk - Govind Puri	82%	79%	51%

VII. Acknowlegdements

The authors would like to acknowledge the support provided by DST-CEFIPRA for the project DST-INRIA 2013-01, ITRA project, funded by DEITy, Government of India, under grant with Ref. No. ITRA/15(57)/Mobile/HumanSense/01, and the H2020 CHOReVOLUTION project.

References

[1] G. Siegemund, V. Turau, and K. Maamra, "A self-stabilizing publish/subscribe middleware for wireless sensor networks," in *Networked Systems (NetSys), 2015 International Conference and Workshops on*. IEEE, 2015, pp. 1–8.

[2] S. Murata and C. Goodman, "Optimally regulating disturbed metro traffic with passenger inconvenience in mind," 1998.

[3] F. Leurent, "On seat congestion, passenger comfort and route choice in urban transit: a network equilibrium assignment model with application to Paris," in *Proceedings of the 88th Annual Meeting of the TRB CD Rom edition*, 2009.

[4] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *Pervasive computing*. Springer, 2004, pp. 1–17.

[5] S. Garg, P. Singh, P. Ramanathan, and R. Sen, "Vividhavahana: smartphone based vehicle classification and its applications in developing region," in *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2014.

[6] E. Miluzzo, C. T. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, and A. T. Campbell, "Darwin phones: the evolution of sensing and inference on mobile phones," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, 2010.

[7] N. Brown, "Edward T. Hall: Proxemic Theory, 1966," *Center for Spatially Integrated Social Science. University of California, Santa Barbara. http://www. csiss. org/classics/content/13 Read*, vol. 18, p. 2007, 2001.

[8] G. P. Perrucci, F. H. Fitzek, G. Sasso, W. Kellerer, and J. Widmer, "On the impact of 2G and 3G network usage for mobile phones' battery life," in *Wireless Conference, 2009. EW 2009. European*. IEEE, 2009, pp. 255–259.

[9] N. Georgantas, G. Bouloukakis, S. Beauche, and V. Issarny, "Service-oriented Distributed Applications in the Future Internet: The Case for Interaction Paradigm Interoperability," in *European Conf. on Service-Oriented and Cloud Computing*, 2013.

[10] R. Racic, D. Ma, and H. Chen, "Exploiting MMS vulnerabilities to stealthily exhaust mobile phone's battery," in *Securecomm and Workshops, 2006*. IEEE, 2006, pp. 1–10.

[11] "Machine learning using MATLAB," http://in.mathworks.com/machine-learning/".

[12] A. Field, *Discovering statistics using IBM SPSS statistics*. Sage, 2013.